НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: «Програмний модуль формування тестів для комбінаційних схем»

Виконав:
студент IV курсу, групи КВ-51
Литвиненко І.В. _____

Керівник:
Доцент кафедри СПіСКС, к.т.н., доцент,
Потапова К.Р. _____

Консультант з нормоконтролю:
Доцент кафедри СПіСКС, к.т.н., доцент,
Клятченко Я.М. _____

Рецензент:
Професор кафедри ОТ ФІОТ, д.т.н.
Сергієнко Я.М. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Пояснювальна записка
до дипломного проекту

на тему: «Програмний модуль формування тестів для комбінаційних схем»

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.П. Тарасенко
«___» червня 2019 р.

ЗАВДАННЯ

на дипломний проект студента

Литвиненко Ігнатій Вадимович

1. Тема проекту «Програмний модуль формування тестів для комбінаційних схем», керівник проекту Потапова Катерина Романівна к.т.н., доцент, затверджені наказом по університету від 22.05.2019 р. №1330-С

2. Термін подання студентом проекту « » червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - Аналіз існуючих рішень та обгрунтування теми дипломного проекту
   - Методи та алгоритми
   - Опис програмного модуля

5. Перелік обов'язкового графічного матеріалу:
   - Міжмодульна взаємодія. Схема структурна.
   - D-алгоритм. Схема алгоритму.
   - Побудова D-кубів. Схема алгоритму.
   - Визначення сигналів на елементі. Схема алгоритму.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятчено Я.М.., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	17.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Аналіз існуючих рішень	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення програмного забезпечення	03.02.2019	
6.	Відлагодження програмного продукту	10.02.2019	
7.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
11.	Підготовка графічної частини дипломного проекту	11.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент І.В. Литвиненко

Керівник проекту К.Р. Потапова

Анотація

Метою дипломного проекту є створення програмного модуля який формує тестові набори для комбінаційних схем. Завдання вирішене у форматі веб-додатку для зручності використання і можливості постійного доступу до сервісу в режимі он-лайн.

В роботі розглянуті і проаналізовані існуючі методи і алгоритми пошуку несправностей в комбінаційних схемах, а також різні варіанти побудови тестових векторів і наборів. Створено та реалізовано програмно алгоритм знаходження тестових векторів (тестовий набір) для мінімізованих комбінаційних схем.

Вхідними даними для даного програмного продукту є таблиця істинності заданої схеми і обраний елемент з вказаною несправністю.

В дипломному проекті реалізовані: алгоритм взаємодії з користувачем, процедури обробки даних і взаємодії блоків, а також програмний алгоритм пошуку тестових наборів.

Ключові слова: D-алгоритм, Схема, Тестові набори, Таблиця істинності, Вектор, D-куб, Алгоритм, Замикання, Несправність, Булева функція, Сигнал, Логічні елементи, Алгебра Рота, Аналіз, Вхід, Вихід.

ANOTATIONS

The purpose of this graduation project is to create a software module that generates test kits for combinational circuits. The task is implemented in the format of a web application for ease of use and the possibility of permanent access to the service in the on-line mode.

The paper reviews and analyzes existing projects, methods and algorithms for troubleshooting in combinational circuits, as well as various options for constructing test vectors and sets. The algorithm for finding test vectors (test set) for minimized combinational circuits was created and implemented in software. The input data for this software product is the truth table of the specified scheme and the selected element with a fault.

In this thesis project was developed: an algorithm for user interaction, a procedure for data processing and interaction of blocks, as well as a programmatic algorithm for finding test sets.

Keywords: D-Algorithm, Scheme, Test Kits, Truth Table, Vector, D-Cube, Algorithm, Closure, Malfunction, Boolean Function, Signal, Logical Elements, Root Algebra, Analysis, Input, Output.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.002 ТЗ	Програмний модуль	4		
			формування тестів для			
			комбінаційних схем			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Програмний модуль	2		
			формування тестів для			
			комбінаційних схем			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Програмний модуль	50		
			формування тестів для			
			комбінаційних схем			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			D-алгоритм.			
			Схема алгоритму.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Литвиненко І.В.			Програмний модуль формування тестів для комбінаційних схем. Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Консульт.						КПІ ім. Ігоря Сікорського,ФПМ,КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.006 Д2	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Міжмодульна взаємодія.			
			Схема структурна.			
	А4	ІАЛЦ.045440.007 Д3	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Побудова D-кубів.			
			Схема алгоритму.			
	А4	ІАЛЦ.045440.008 Д4	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Визначення сигналів			
			на елементі.			
			Схема алгоритму.			
	А4	Презентація	Слайди презентації	10		

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмний модуль формування тестів для комбінаційних схем».

Галузь застосування: тестування роботи логічних схем

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення web-додатка для пошуку тестових наборів в комбінаційних схемах.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою (Windows,Linux, Mac OS,IOS,Android,WP);
- можливість вибору елементу для тетування;
- можливість вибору помилки.

5.2. Вимоги до апаратного забезпечення

- Процесор: 2,4-ядерний;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet.

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Linux, Mac OS, Windows Phone 8, Android, iOS;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.004 ПЗ	Програмний модуль	50		
			формування тестів для			
			комбінаційних схем			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			D-алгоритм.			
			Схема алгоритму.			
	А4	ІАЛЦ.045440.006 Д2	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Міжмодульна взаємодія.			
			Схема структурна.			
	А4	ІАЛЦ.045440.007 Д3	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Побудова D-кубів.			
			Схема алгоритму.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
			прин				
Розробив		Литвиненко І.В.			Мікропроцесорна система дистанційного моніторингу параметрів обладнання. Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Програмний модуль	1		
			формування тестів для			
			комбінаційних схем			
			Визначення сигналів			
			на елементі.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			
	А4	Презентація	Слайди презентації	10		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

КС – комбінаційна схема.

ТВ – тестовий вектор.

ТН – тестовий набір.

ЛС – логічна схема.

C# – мова програмування високого рівня.

ПЗ – програмне забезпечення.

КЗ – коротке замикання.

ЕОМ – електронно обчислювальна машина.

ВСТУП

У практиці проектування ЕОМ накопичений величезний досвід по синтезу різних схем. Такі пристрої як дешифратори, шифратори, схеми порівняння, комбінаційні суматори, комутатори та ін. є комбінаційними схемами, застосування яких дуже розвинене в сучасному проектуванні цифрових пристроїв. Правильне функціонування цифрового пристрою можливо тільки в тому випадку, якщо кожен елемент схеми працює без спотворення сигналу.

Основною метою тестового діагностування цифрових схем на рівні комбінаційної схеми, друкованої плати або цифрової системи, є виявлення несправностей, що обумовлені дефектами, здобутими в процесі виробництва, експлуатації або в результаті старіння. Другою метою тестування є визначення місця і причини дефекту з достатньою точністю і достовірністю. Цей вид тестового діагностування включає в себе як перевірку на правильність роботи, так і локалізацію місця несправності. Основною ідеєю побудови тестового вектору є подача на входи таких сигналів, щоб по результату на виході зі схеми можна було б однозначно визначитися з помилкою та місцем її знаходження.

Тестування комбінаційної схеми є надзвичайно важливим для забезпечення високого рівня функціональності. Розмір цифрових схем та щільність компонування постійно збільшується , це впливає на вартість і час для тестування. Для аналізу такого контуру потрібна висока якість тесту з мінімальною кількістю комбінацій введення.

У цій роботі аналізуються відомі програмні продукти які займаються аналізом й тестуванням комбінаційних схем, підкреслюються їх плюси й мінуси. Реалізовано програмний модуль для пошуку мінімальної кількості вхідних тестових векторів для виявлення несправності, як, наприклад, константні 0 чи 1.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

В останні роки розвиток технології інтегральних схем швидко зростає. Цифрові системи будуються з усе більшою і більшою складністю, перевірка несправностей і діагностика цифрових схем набирає величезної ваги і стає невід'ємною частиною виробничого процесу. З розвитком науки і техніки сучасні пристрої стають все більш і більш складними з кожним днем. У міру збільшення складності пристрою, тестування стає ще більш складним. Схеми зменшуються в фізичному розмірі, а також збільшуються як в швидкості, так і в діапазоні можливостей. Це призводить до збільшення часу випробувань і зростанню вартості тестування, що спонукає до розробки програмного забезпечення для швидкого й надійного тестування.

Нажаль на сьогоднішній день існує зовсім не багато програмного забезпечення яке знаходиться у відкритому доступі и дає можливість швидкого формування тестів для комбінаційних схем. Проаналізуємо та розглянемо можливості існуючих продуктів:

- EveryCircuit
- Multisim (National Instruments)
- LTspice
- DoCircuits
- Circuit Lab
- TINA

EveryCircuit

Для розробки EveryCircuit спроектований движок моделювання, оптимізований для інтерактивного мобільного використання. Реалізує чисельні методи та реалістичні моделі пристроїв. Бібліотека компонентів дає вам можливість створювати будь-які аналогові або цифрові схеми.

Можливий процес тестування та налагодження.

Переваги:

- надає користувачу можливість конструювати схеми в режимі on-line;
- можливість роботи на ПК й на Android та iOS.

Недоліки:

- платний доступ до функціоналу;
- не інтуїтивне налаштування.

Візуальна складова EveryCircuit на найвищому рівні серед програм, що займаються побудовою й обробкою схем. Розробники намагалися зробити максимально красивим свій проект (рис. 1.1).

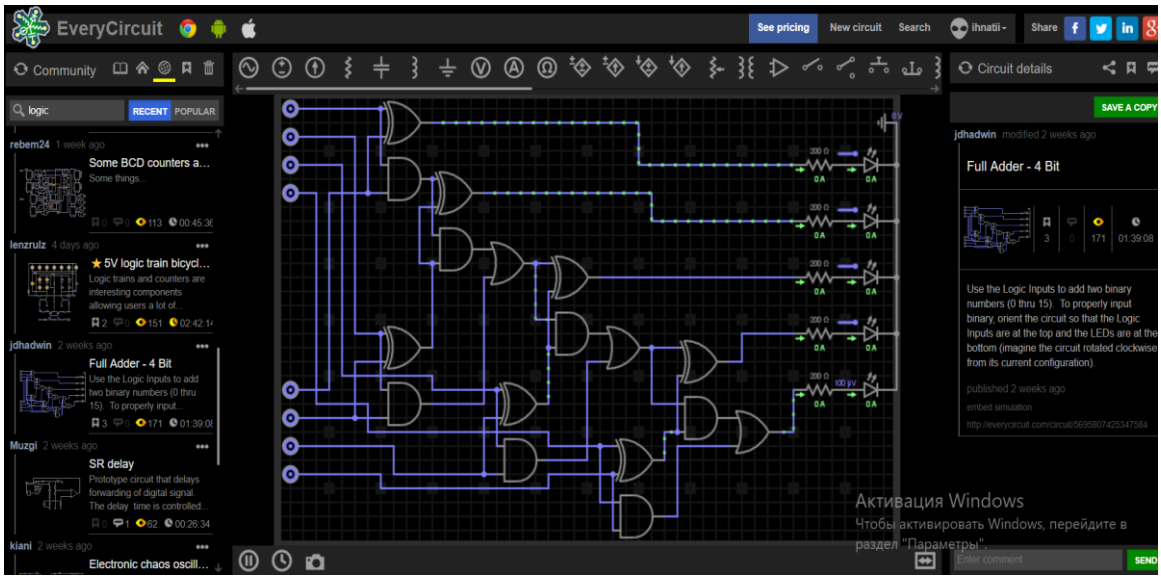


Рисунок 1.1 – Інтерфейс EveryCircuit

На схемі ми можемо побачити й підсвічені шляхи проходження сигналу й переходи з одного стану в інший. Якщо ви моделюєте схеми, наприклад, з вимірювальною технікою (вольтметр, амперметр и т.д.) або з регульованими блоками подачі живлення, в вас є можливість побачити всі зміни на схемі як в реальному житті. Але таке розташування пріоритетів призвело до виведення продукту на рівень лише початківців. Також є платна версія але навіть це,

особливо нічого не змінює. Хоча розробники обіцяють випустити нову версію де буде розширений функціонал й тестові можливості.

Значною перевагою цього додатку є постійний доступ до нього. Це дає можливість користуватися в будь-якому місці й в будь-який час, що дуже подобається студентам.

Multisim (National Instruments)

Один з найбільш відомих на ринку інструментів для проектування и моделювання схем. Він включає в себе версію Multicap, що робить його універсальним засобом для негайного подальшого тестування схем.

Переваги:

- великий функціонал;
- моделювання схем будь-якої складності;
- велика бібліотека.

Недоліки:

- платний доступ до розширеної версії;
- режим знаходження тестових векторів лише у платній версії.

NI Multisim - єдиний в своєму роді продукт. Він володіє повнофункціональної структурою схемотехніки, яка настільки універсальна, наскільки потребує ваша поставлена задача. Має простий, інтуїтивний інтерфейс (рис. 1.2).

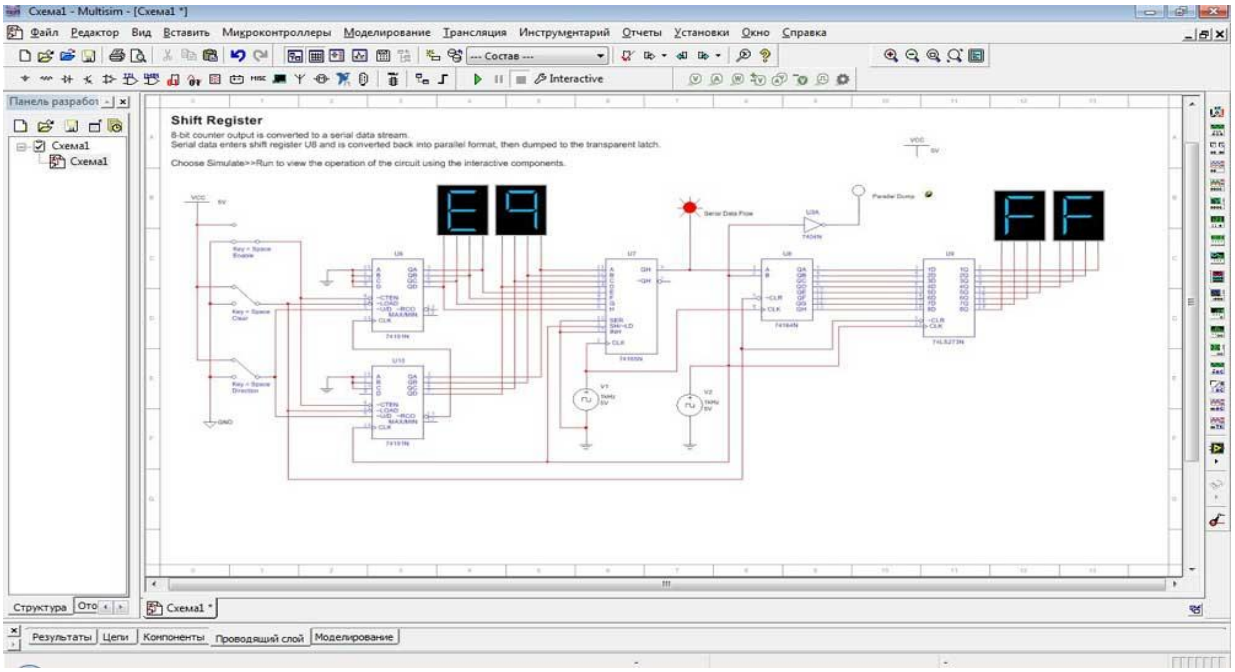


Рисунок 1.2 – Інтерфейс Multisim

Головним мінусом програми є те, що доступ до розширеного функціоналу, та великої кількості бібліотек є тільки в тих користувачів як можуть собі дозволити сплатити 1000 доларів за базову версію й 4000 доларів за саму просунуту версію. Це надзвичайно дорого. В Multisim все ще пропонують ознайомчу версію, але в ній кількість бібліотек (і доступних електронних символів) обмежена.

LTspice

LTspice IV - це безкоштовний програмний симулятор реалізації електронних схем, створений виробником лінійних технологій (LTC). Це високопродуктивний симулятор SPICE, що реалізує схему захоплення і перегляду форми хвилі з поліпшеннями і моделями для спрощення моделювання регуляторів (рис. 1.3). Удосконалення SPICE зробили моделювання регуляторів надзвичайно швидким в порівнянні зі звичайними імітаторами SPICE, що дозволяє користувачеві переглядати хвильові форми для більшості перемикаючих регуляторів всього за кілька хвилин, в цю

завантаження включені 1 spice 4, макромоделі для 80% лінійних регулятори технології перемикання, більше 200 моделей операційних підсилювачів, а також резистори, транзистори та моделі Mosfet.

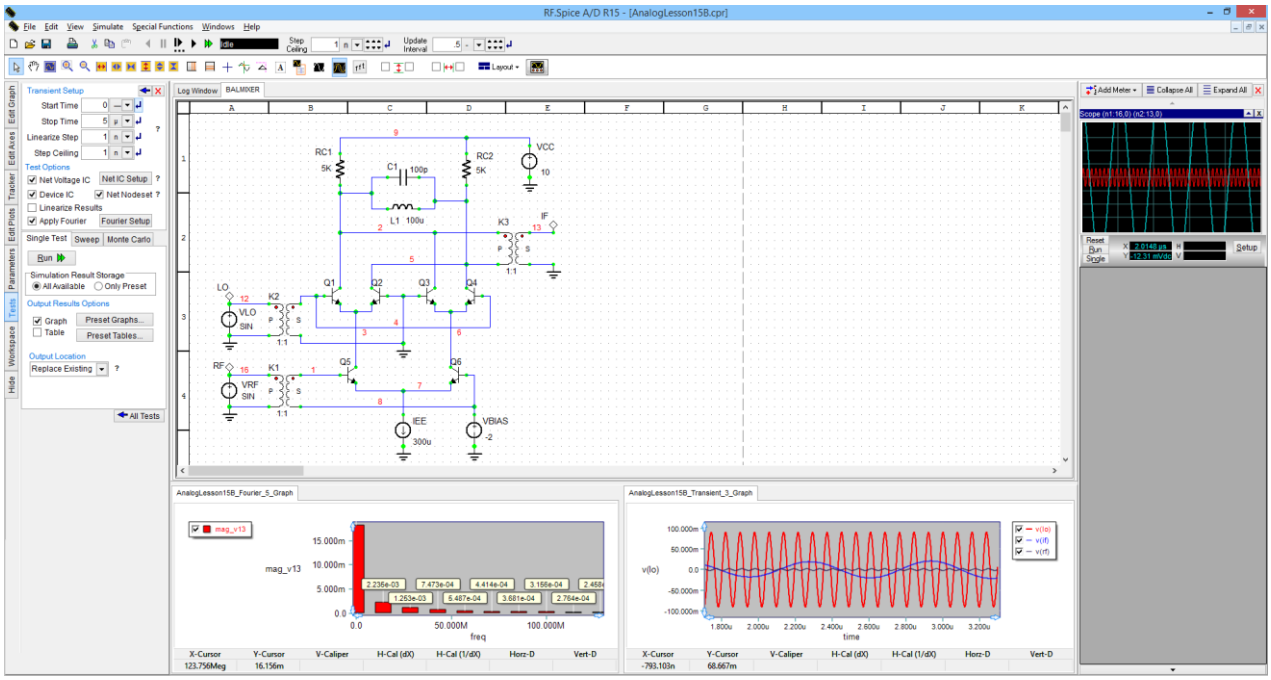


Рисунок 1.3 – Розробка в LTspice

Весь наведений вище аналіз можна змоделювати за допомогою LTspice. У чому сенс подібної симуляції? Суть такого рішення заключається у тому, що надається можливість відтворити репліку реального середовища за допомогою доступних інструментів для необхідного лабораторного або тестового середовища.

Переваги використання LTspice:

- безкоштовне програмне забезпечення для проектування і аналізу схем. Це набагато дешевше, ніж купувати багато реальних запчастин, справжній генератор сигналів і осцилограф;
- аналіз постійного струму, частотний відгук, аналіз n-портів, набагато простіше симулювати в LT Spice, ніж теоретично аналізувати;
- моделювання стабільного ланцюга з необмеженою кількістю вузлів, редактор схем / символів, засіб перегляду форми хвилі, бібліотека

пасивного пристрою

Недоліки:

- великі сумніви щодо зручності при моделюванні великих складних конструкцій;
- потрібно багато зусиль для отримання відповідних даних для моделювання та інтерпретації вихідних даних;
- складність перевірки правильності моделей.

DoCircuits

Один з найбільш базових інструментів для проектування і моделювання схем в мережі. DoCircuits дозволяє тестувати і вимірювати з реальним лабораторним обладнанням і ділитися своїм дизайном.

Переваги:

- зручний функціонал;
- доступ on-line;
- інтерактивний дизайн.

Недоліки:

- погана підтримка розробників
- не безкоштовний доступ;
- не знаходить тестові вектори.

Програмне забезпечення при схемному моделюванні використовує математичні моделі для відтворення поведінки реального електронного пристрою або схеми. Завдяки своїй високій точності моделювання, він широко використовується у навчальних цілях в технічних університетах.

Продукт привертає дуже багато студентів, інтегруючи їх у навчальний процес. Долучаючись до побудови, синтезування, налагодження й аналізу своїх проектів, учні дуже сильно поглиблюють свої знання.

DoCircuits - це хмарне додаток, на розробку якого пішло близько дев'яти місяців. Користувачі заходять на www.DoCircuits.com і створюють логін. Один логін дозволяє їм створювати незліченні схеми (рис. 1.4). Натиснувши «Launch Virtual Labs», вони можуть зберегти ці схеми в хмарі, щоб до них можна було отримати доступ з будь-якого місця. Це дозволяє користувачам ділитися своїми схемами з іншими.

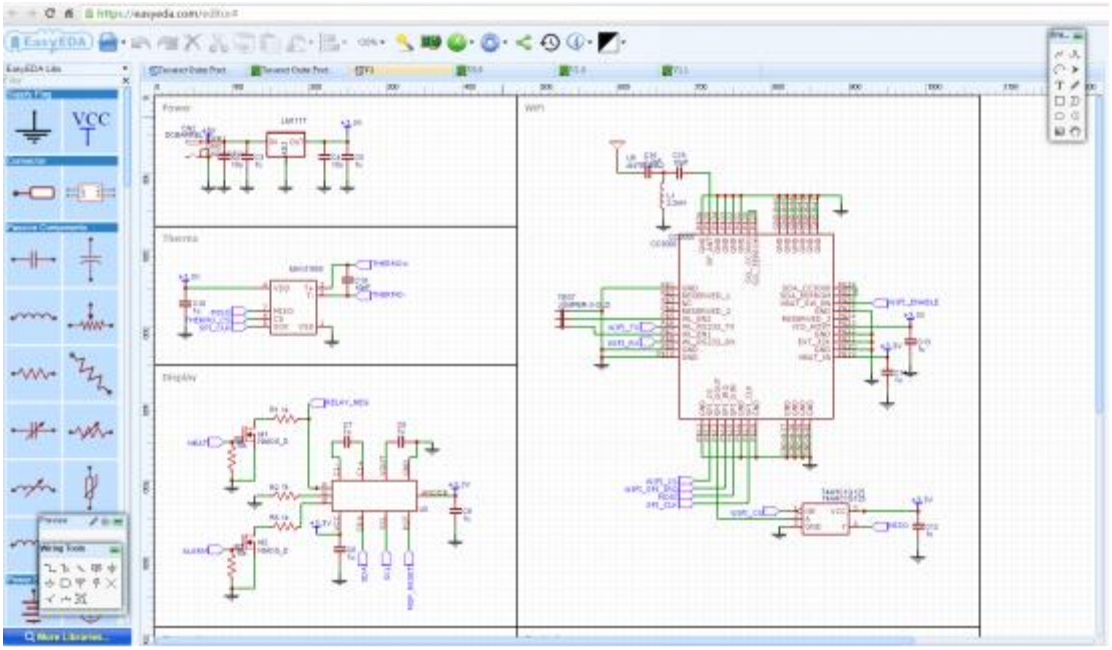


Рисунок 1.4 – Інтерфейс в DoCircuits

Коли користувач створює схеми і запускає їх в хмарі, результати обробляються і показуються користувачеві для аналізу. Хмарний бекенд реалізований в Microsoft Azure, а інтерфейс - у Flex. Ви можете отримати доступ до функцій моделювання, навіть не входячи в систему. Але щоб використовувати функції спільного використання і збереження, необхідно увійти в систему.

Значним недоліком цієї програми є те, що не має інструментів для формування тестових наборів в комбінаційних схемах. Також розробники цього продукту зовсім не хочуть йти на контакт й впроваджувати якісь покращення які висловлюють користувачі.

TINA

Переваги:

- різні типи схем можна моделювати;
- реалізація найскладніших моделей.

Недоліки:

- не безкоштовна;
- не для початківців.

TINA Design Suite - це потужний, але доступний програмний пакет для аналізу, проектування і тестування в реальному часі аналогових, цифрових, мікроконтролерів і змішаних електронних схем і схем їх друкованих плат. Продукт дає можливість схеми аналізувати, тестувати і налагоджувати.

Введіть і проаналізуйте будь-яку схему на 100 вузлів (студентська версія) або 200 (базова версія) протягом декількох хвилин за допомогою простого у використанні редактора схем TINA (рис. 1.5). Поліпшите свої схеми, додавши текст і графіку. Виберіть компоненти з великої бібліотеки, яка містить понад 10 000 моделей виробників. Існує 20 різних режимів та 10 віртуальних інструментів для легкого та швидкого аналізу вашої схеми.

Налаштуйте презентації, використовуючи передові інструменти малювання TINA для управління текстом, шрифтами, осями, шириною лінії, кольором і макетом. Ви можете створювати і друкувати документи безпосередньо всередині TINA або вирізати і вставляти результати в ваш текстовий редактор або DTP-пакет.

Одним з недоліків платформи TINA є платний доступ до ресурсу навіть для студентів, хоча він значно дешевший за Multisim. Студентська версія коштує 50 доларів. Також, щоб працювати з продуктом на повну потрібно бути досить досвідченим у галузі проектування та тестування схем.

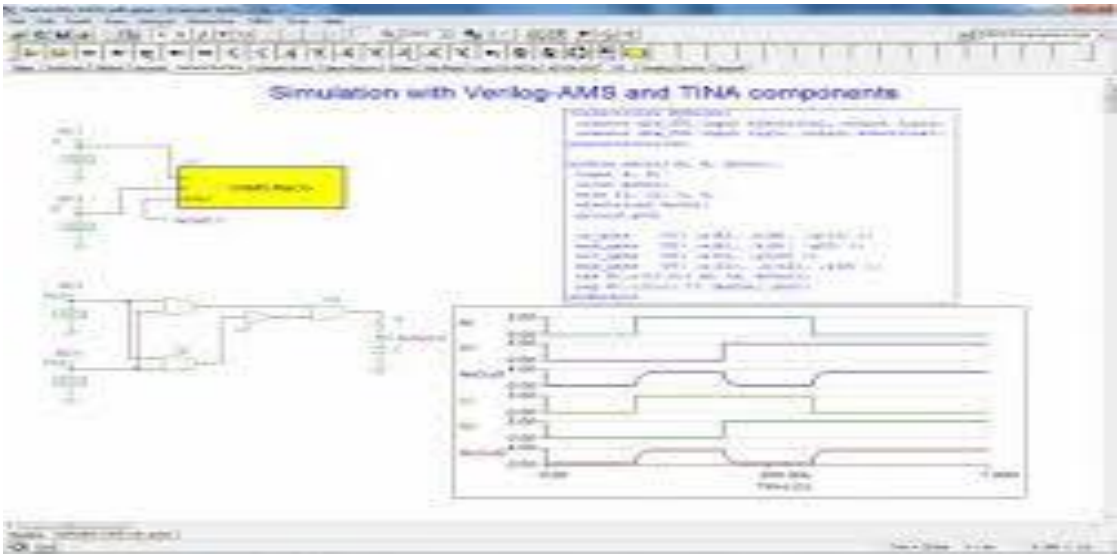


Рисунок 1.5 – Інтерфейс TINA

Проаналізувавши значну частину ринку додатків, для побудови, досліджень та тестування схем, можемо сказати, що у відкритому доступі існує дуже мало продуктів які повністю задовільнили б наші умови. У більшості випадків в основних функціях є: побудова схеми, аналіз, проведення замірів, діагностика, побудова графіків. Виявлення тестових наборів не зустрічається майже в жодному проекті а там де є, доступ до функціоналу коштує нереальні гроші. Моєю головною задачею було, виправити це й реалізувати знаходження тестових векторів які будуть однозначно вказувати на те, що у елементі схеми, дійсно є та чи інша проблема. Щоб реалізувати цю задачу мені потрібні лише таблиця істинності та вибраний користувачем елемент комбінаційної схеми.

У цій роботі спроектовано та розроблено програмне забезпечення яке включає основні позитивні положення вище зазначених продуктів. Також була врахована основна частина недоліків таких програм:

- немає безкоштовного доступу;
- погана підтримка;
- немає доступу з будь-якого пристрою з браузером.

Проект являє собою модуль яким можна доповнити кожен з проектів працюючих з моделюванням або аналізом комбінаційних схем.

2. МЕТОДИ ТА АЛГОРИТМИ

2.1 Несправності та їх пошук

Перш ніж почати пошук несправності схеми, бажано отримати хоча б загальне уявлення про те, які з них можуть виникнути в цьому пристрої. Існує досить багато різних типів фізичних несправностей, що впливають на роботу електронних схем. Щоб спростити тестування, зручно змоделювати можливі помилки. У загальному випадку модель може будуватися на ефекті, який дана неисправність надає на схемі. Несправності можна розділити на дві великі категорії: логічні моделі несправностей і моделі параметричні. Перші описують несправності, які впливають на логічну функцію схеми. Другі - несправності, які змінюють параметри схеми, наприклад величину напруги, струму, електричного навантаження. У цьому розділі будуть розглядатися тільки логічні несправності.

Константні несправності.

Багато несправності можна промоделювати, вважаючи що вони призводять до появи на сигнальній лінії – фактично на вході або виході вентиля - постійного логічного рівня напруги (0 або 1). Несправність, що призводить до встановлення на лінії постійного сигналу носить назву константної ("залипання" в 0 або в 1).[7]

Інші несправності.

Крім ситуацій, коли на лінії примусово з'являється постійний логічний рівень 1 або 0, часто виникають інші збої в роботі. Іноді внутрішнє коротке замикання або розрив ланцюга можуть призводити не до появи постійного 0 або 1, а до виникнення на виході деякого проміжного рівня логічного сигналу. Такий тип несправностей називається константою несправністю з залипання в

відкритому стані, або константної несправністю транзисторного рівня. Знайти такі неполадки завжди важко, і тут вони розглядатися не будуть.

Коротке замикання між двома частинами схеми так і називається: несправність типу замикання, або перемичка. Вплив КЗ на вентиль залежить від технології виготовлення останнього і місця розташування. Може виявитися, що буде змінена булева функція або сигнал на лінії "Залипне" на постійному рівні, а можуть виникнути і інші помилки функціонування схеми. Наприклад, замикання між входом і виходом комбінаційної логічної схеми може призвести до формування послідовної схеми. Створення перемички між двома виходами може викликати зміну рівня напруги в лінії, причому величина цього напруги буде залежати від відношення потужностей різних вентилів і їх здатності керувати сигналом в лінії.[9]

Константна несправність.

Модель константної несправності описує вплив фізичної несправності на вхідні / вихідні сигнали вентиля. Для вентиля І з двома входами існує шість можливих константних несправностей - по одній з двох кожного типу для кожної з трьох ліній вентиля. Для k сигнальних ліній існує 2k різних комбінацій потенційних константних несправностей. При виникненні множинних несправностей кожна сигнальна лінія може перебувати в трьох станах: бути вільною від неполадок, перебувати в стані з постійним рівнем логічного 0 і в стані з постійним рівнем логічної 1. Для k різних сигнальних ліній є всього 3^k різних комбінацій трьох станів. З них тільки одне вільне від помилок, звідси, існує 3^k - 1 різних станів з несправностями. Таким чином, число комбінацій помилок може бути дуже велике. У цій роботі будуть розглядатися тільки одиночні несправності, оскільки аналіз множинних несправностей в явному вигляді - завдання досить складне. На щастя, переважна більшість множинних несправностей знаходиться під час пошуку одиночних помилок.[7] На рис. 2.1.1 показана найпростіша логічна схема, у якій три

початкові входи, A, B і C, і один вихід зі схеми. У ній також є всього одна внутрішня лінія. Проблема може проявитися на будь-який з сигнальних ліній схеми, тому на малюнку хрестиками показані п'ять можливих ділянок їх появи: лінії A, B, C, f, x. Сигнали на кожній з цих ліній можуть перейти в стан з постійним логічним рівнем - 0 або 1. Отже, всього в цій схемі можливі 10 різних константних несправностей.

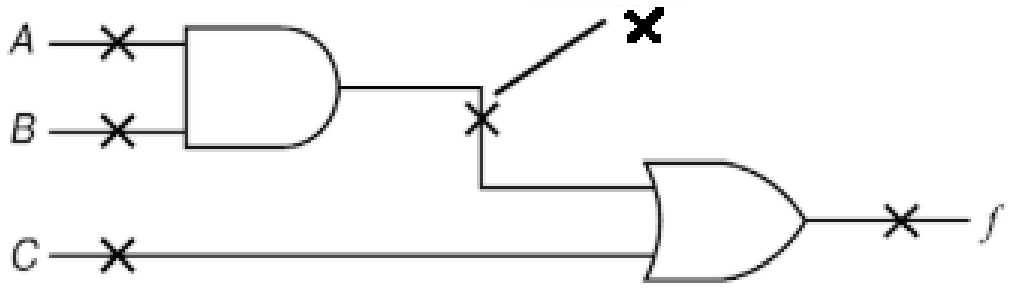


Рисунок 2.1.1 – Логічна схема

Якщо вихід вентиля з'єднаний з декількома входами наступного вентиля, то кожен шлях проходження сигналу необхідно розглядати окремо, так як модель константної несправності є відображення внутрішньої неполадки, яка, можливо, не є відмовою саме сигнальної лінії, що впливає на всі входи, підключені до даного виходу. На рис. 2.1.2 показана схема, на якій вихідний сигнал одного вентиля управляє входами двох інших. У цьому випадку може виникнути відразу вісім різних константних несправностей. Тут проблема на виході вентиля І може привести до "залипання" рівнів на входах обох наступних вентилів АБО і НЕ. Ці вентилі також можуть мати константні несправності на входах, але вони вже впливатимуть на функціонування тільки їх самих.

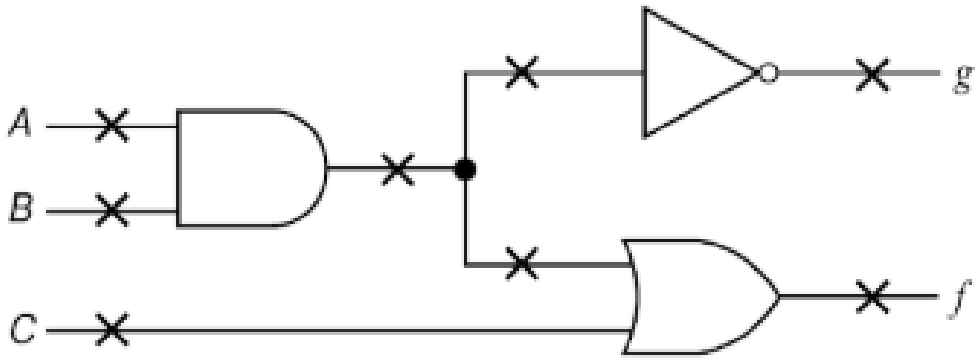


Рисунок 2.1.2 – Комбінаційна схема

2.2 Алгоритми знаходження тестових наборів

Найпростіші схеми легко протестувати, подаючи на всі входи певні значення рівнів і порівнюючи вихідні сигнали з таблицею істинності схеми. Наприклад, щоб провести повне тестування вентиля І з двома входами, достатньо подати на них кодові комбінації 00, 01, 10 і 11 і побачити, що вихідні значення рівні, відповідно, 0, 0, 0 і 1. Якщо ж хоч в одному випадку значення на виході відрізняються від очікуваного, то це означає, що в схемі вентиля є несправність. Вхідні комбінації, які використовуються для знаходження несправностей, звуться тестові вектори. Тестові вектори, які подаються на схему для перевірки її працездатності, називаються тестовим набором.[6] Наприклад, вичерпні випробування двух-входового вентиля І полягають в подачі на цей елемент тестових векторів 00, 01, 10 і 11, що складають тестовий набір. Кількість комбінацій, які необхідно подати на входи схеми для її повного тестування, залежить, звичайно ж, від числа входів. Для n входів є 2n різних вхідних комбінацій. Отже, зі збільшенням складності схеми і кількості входів подавати на них всі можливі комбінації вхідних сигналів стає непрактично. Покриттям несправностей тесту називається відносне число помилок, які з його допомогою можна виявити.[8]

Основною проблемою даного способу є знаходження мінімальної кількості тестових векторів при за-даному покритті несправностей.

Табличний і алгебраїчний методи

Якщо відома булева функція схеми, то порівняння функції з помилкою внаслідок несправності і очікуваної функції дозволить отримати булеві вирази для тестових векторів. Це можна зробити, порівнюючи таблиці істинності, в яких є значення вихідних сигналів при наявності помилки і без неї. Маючи булеву функцію, можна також шляхом виконання визначених алгебраїчних маніпуляцій, домогтися отримання тестового набору. Тут ці методи розглядатися не будуть, тому що вони підходять тільки для пошуку несправностей в невеликих схемах, робота яких описується простими логічними рівняннями.[10]

Метод активізації шляху

Метод активізації шляху підходить для більш складних комбінаційних логічних схем і складається з перевірки несправності в заданій точці шляхом подачі в неї значення сигналу, протилежного тому, яке викликається несправністю. Якщо тестується несправність типу "залипання в 1", то в точку несправності подається 0. Якщо ж тестується константна несправність з постійним низьким рівнем ("залипання в 0"), то в точку подається 1. За рахунок подачі на вхід сигналів з відповідними рівнями активізується шлях передачі логічного рівня сигналу в точці несправності крізь схему до виходу, де його можна спостерігати. Це яскраве представлення D-алгоритм, який буде описано в пункті 2.3.[1]

1) Спосіб сенсибілізації шляху.

Це один з найбільш ранніх методів, що використовуються для виявлення несправностей. У цьому методі тест виявлення несправностей може бути

знайдений шляхом вивчення шляхів передачі від місця прийнятої несправності до одного з його виходів. Це принципова ідея методу сенсибілізації шляху. Метод є дуже привабливим з точки зору того, що не вимагає побудови таблиці несправностей, і є корисним для виявлення несправностей ланцюгів типу дерева. Але для загальних не малих схем процес вичерпання всіх можливих одиночних контурів, передбачає досить багато пошуку і обчислень, навіть коли це робить комп'ютер. Коли існує розгалуження, виникає додаткова проблема сенсибілізації набору шляхів, які фактично містять всі з'єднання. Було б бажано, щоб такі випробування були знайдені шляхом безпосереднього огляду схеми. На жаль, такої прямої методики не було виявлено.[6]

2) Метод еквівалентно-нормальної форми.

Еквівалентна Нормальна Форма (ENF) схеми отримується шляхом вираження кожного виходу як суму його входів і збереження ідентичності кожного входу и виходу відповідним індексом. Кожна вхідна проіндексована змінна в ENF називається літералом. Еквівалентна нормальна форма відповідає дворівневій схемі AND-OR. Отже, методи, розроблені раніше для дворівневих схем, тепер можуть застосовуватися з кількома модифікаціями до еквівалентних нормальних форм. Однак між цими двома типами схем існують дві основні відмінності. У схемі AND-OR кожен літерал відповідає входу схеми і унікальному шляху від цього входу до виходу схеми. З іншого боку, в еквівалентній нормальній формі дві змінні можуть мати різні індекси, оскільки вони пов'язані з двома різними шляхами, хоча вони відповідають одному і тому ж входу в оригінальній багаторівневій схемі.[2]

Метод сенсибілізації шляху та метод еквівалентної-нормальної форми базуються на концепції сенсибілізації шляху. Метод ENF має ряд переваг за методом сенсибілізації шляху. Метод ENF є аналітичним методом, який забезпечує засоби для систематичного пошуку найбільш бажаних випробувань, кожен з яких виявляє багато несправностей у контурі. Одиночні

шляхи, які не є сенсибілізуючими, зазвичай легко бачити з ENF. Спосіб отримання тестів за допомогою ENF дуже простий.

3) Виявлення несправностей на двох рівнях.

Попередні методи побудови тестів для виявлення помилок в комбінаційних схемах базувалися на дослідженні кожного шляху (метод сенсибілізації шляху і метод ENF). Третій підхід до проблеми, замість розгляду кожної окремої несправності або кожного шляху пропонує, розглядати кожен елемент схеми. Представлено дуже простий і прямий метод побудови мінімального повного тесту на виявлення несправностей для будь-якого дворівневого І-АБО (АБО-І, АБО-АБО тощо), що використовує цей підхід.[10]

Цей метод може бути представлений двома методами - графічним й табличним. Графічна версія, яка використовує карту Карно, застосовується до схем з невеликою кількістю вхідних змінних, скажімо, не більше шести, але переважно не більше чотирьох. Потім, як і те, що було зроблено при мінімізації функцій перемикання, ця версія карти Карно розширена до табличного метода, аналогічно тому, як у Квайна Мак-Класки. Він використовує точно ті ж принципи, але без карт, дозволяє схемі мати будь-яке кінцеве число вхідних змінних, тобто є дуже привабливим з точки зору машинного обчислення.[1]

4) Метод булевої різниці.

Ідея використання булевої різниці для аналізу помилок була запропонована Е.Л. Бернсоном в 1965 році. У 1967 році була опублікована стаття, що описує подібні ідеї. Здається, що засновником різницевих методів був Г. Буль, хоча він не застосовував їх до булевих рівнянь.

Булева різниця визначається як операція між двома булевими функціями, одна з яких представляє нормальну схему а інша - несправну. Таким чином,

якщо логічна різниця 1, позначається помилка. Припустимо, що є функція перемикання, яка має один вихід F і n входів x1, x2,… .xn, так що F (X) = F (x1, x2,…., Xn). Якщо один з входів у функцію перемикання був помилковим, скажімо, вхід x`i, то вихід буде F (x1,… .., x'i, ……, xn). Для аналізу дії схеми, коли помилка відбувається, треба знати, за яких обставин два виходи однакові.[3]

5) Цілочисельний лінійний метод програмування.

У цілочисельному методі лінійного програмування для заданої комбінаційної схеми генеруються дві таблиці. Вони є таблицею несправностей (табл. 2.1.1) і таблицею виявлення несправностей (табл. 2.1.2). На базі таблиці виявлення несправностей формується матриця діагностики несправностей. Діагностична матриця Fjn формується за допомогою рядків тестових номерів й стовпців zf1, zf2 ..zfi, як показано в таблиці (табл. 2.1.3).[1]

Test No	**X_1**	**X_2**	**..**	**X_n**	**Z**	**f_1**	**f_2**	**…**	**f_i**
1	0	0	..	0	0	1	0	…	0
2	0	0		1	1	1	0	…	1
. . .	. . .	. . .	..	. . .	. . .	. . .	. . .	…	. . .
2	1	1	..	1	0	0	0	…	1

Таблиця 2.1.1 – Таблиця несправностей

Test No	**X_1**	**X_2**	**..**	**X_n**	**Z**	**Zf_1**	**Zf_2**	**…**	**Zf_i**
1	0	0	..	0	0	1	0	…	0
2	0	0		1	1	0	1	…	0
. . ·n	. . .	. . .	..	. . .	. . .	. . .	. . .	…	. . .
2	1	1	..	1	0	0	0	…	1

Таблиця 2.1.2 – Таблиця виявлення несправностей

Тестовий номер	Номер помилки						
	1	2	3	4	. . .	. . .	n
1	1	0	0	0	. . .	. . .	0
2	1	0	1	0	. . .	. . .	1
3	0	1	0	1	. . .	. . .	0
.	.	.	.	.	. . .	. . .	.
.	.	.	.	.	. . .	. . .	.
J	0	0	1	0	. . .	. . .	1

Таблиця 2.1.3 – Діагностична матриця

У наведених вище матрицях рядки ідентифікують тестові числа і стовпці, що ідентифікують числа помилок. Припустимо, що комбінаційна ланцюг має n помилок, значення {0,1}присвоєні змінним f1, f2,…, fn. Вектору V з n входами для кожного тестового числа j присвоюється значення {0, 1} для змінних Vj, j = 1, 2,. . .J (де J з 2n), до кожного вектора. Проблема пошуку мінімального тестового набору полягає в тому, щоб знайти найменшу підмножину цих векторів, яка виявляє всі несправності. У цій задачі тестовий набір вибирається наступним чином: вектор j включається у вибраний векторний набір, якщо Vj = 1. Вектор j відкидається у вибраний векторний набір, якщо Vj = 0.[4]

Налаштування несправностей та тестовий набір моделюються без скидання помилок. Результат представлений у вигляді діагностичної матриці 0 і 1, як показано в Таблиці III. У цій матриці елемент Fjn = 1 тільки, якщо помилка n виявлена вектором j. Цей алгоритм мінімізує кількість тестових випадків, які охоплюються максимальним числом несправностей, наявних у контурі. Цей метод може бути застосований для складних схем.

Порівняння вище обговорених методів (табл. 2.1.4).

No	Методи	Переваги	Обмеження
1	Спосіб сенсибілізації шляху	Підходить для не деревоподібних схем. Метод сенсибілізації шляху є дуже привабливим тому, що не вимагає таблиці несправностей.	Процес вивчення всіх можливих шляхів, що відбуваються з виходу воріт, передбачає багато операцій пошуку.
2	Метод еквівалентної норми	Метод ENF визначає певні шляхи, які не є сенсибілізуючими в інших методах.	Метод не гарантує повної діагностики даної схеми, оскільки скоригована методика дає абсолютно різні шляхи для ENF і його доповнення.
3	Дворівневі методи виявлення несправностей	Дуже прості і прямі методи, придатні для будь-якого дворівневого І-АБО (АБО-І, АБО-АБО і т.д. Табличний метод не використовує техніку k-map.	Метод карти Карно вимагає схем з максимум 6 вхідними змінними.
4	Метод булевої різниці	Концептуально прості способи отримання тестових послідовностей для комбінаційних схем.	Складність обчислень зростає з числом вхідних даних.
5	Цілочисельний лінійний метод програмування	Він може бути прийнятий для складних схем. Забезпечується оптимальне рішення.	Складність стрімко зростає.

Таблиця 2.1.4 – Порівняння методів

2.3 D-алгоритм

Обговоримо й проаналізуємо D- алгоритм більш досконало.

Визначення.

Вироджене покриття логічного елемента є мінімальний набор призначень вхідного сигналу. У таблиці 2.3.1 вироджене покриття для елементів І та НЕ-АБО.

Gate type	Inputs		Output	Gate type	Inputs		Output
AND	*a*	*b*	*d*	NOR	*d*	*e*	*F*
1	0	X	0	4	1	X	0
2	X	0	0	5	X	1	0
3	1	1	1	6	0	0	1

Таблиця 2.3.1 – Вироджене покриття

D-куб - це згорнутий запис таблиці істинності, який може бути використаний для характеристики довільного логічного блоку. Об'єднати рядки 3 і 1 виродженого покриття елемента AND та 2 і 1, виразити це через п'ятизначну алгебру Рота (Complete d-intersection operation табл. 2.3.2) і ви отримаєте D-куб поширення.[11]

D	1/0	1	0
$\overline{D}$	0/1	0	1
0	0/0	0	0
1	1/1	1	1
X	X/X	X	X

Таблиця 2.3.2 – Алгебра Рота

Тепер можна зробити повний набір D-кубів, що описують, як несправності поширюється через логічні елементи. Перехід з таблиці виродженого покриття до таблиці D-куба для елемента OR проілюстровано в таблицях 2.3.3 та 2.3.4, а для AND в таблицях 2.3.5 та 2.3.6.

A	B	C
0	0	0
X	1	1
1	X	1

Таблиця 2.3.3- Вироджене покриття

A	B	C
D	0	D
0	D	D

Таблиця 2.3.4 – D-кубів

A	B	C
0	0	0
X	1	1
1	X	1

Таблиця 2.3.5- Вироджене покриття

A	B	C
D	0	D
0	D	D

Таблиця 2.3.6 – D-кубів

Розглянемо приклад реалізації D-алгоритму.

1. Активація помилки

Активуємо помилку на комбінаційні схемі як показано на рисунку 2.3.1.

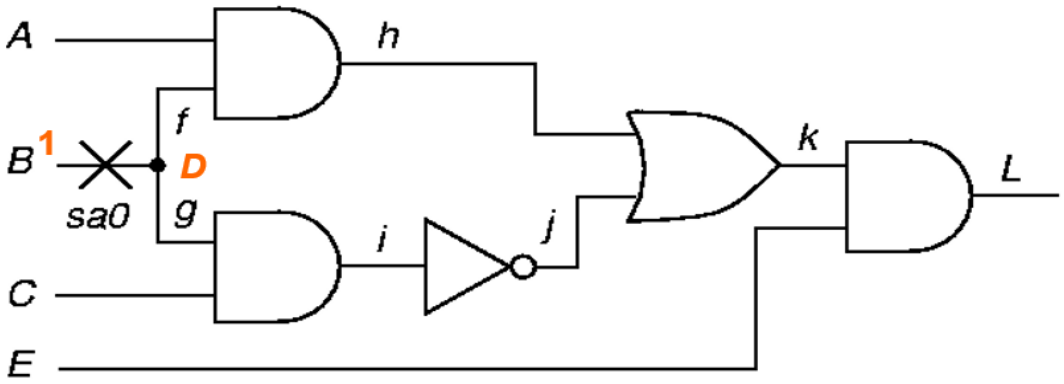


Рисунок 2.3.1 – Активація помилки

Активуємо несправність на лінії g та f. В нашому випадку помилкою буде постійний логічний 0. З цього зрозуміло, що треба подати на вхід B логічну 1, для того, щоб побачити помилку. За п’яти значною алгеброю Рота (таб. 2.3.2) бачимо, що поширювати ми будемо сигнал D. [12]

2. Просування до виходу із схеми

Починаємо рухатися по схемі через кожен елемент, розставляючи значення на виходах і входа спираючись на D-куб кожного з елементів (рис. 2.3.2). Головним принципом проходу цього проходу є доставка D до віходу із схеми. Першим бачимо AND. Одразу ставимо на h – D и по таблиці D-куба визначається вхід A. Переходимо до наступного елемента OR й ставимо на j логічний 0. Аналогічно з першим елементом визначаємо вхід E як логчна 1.

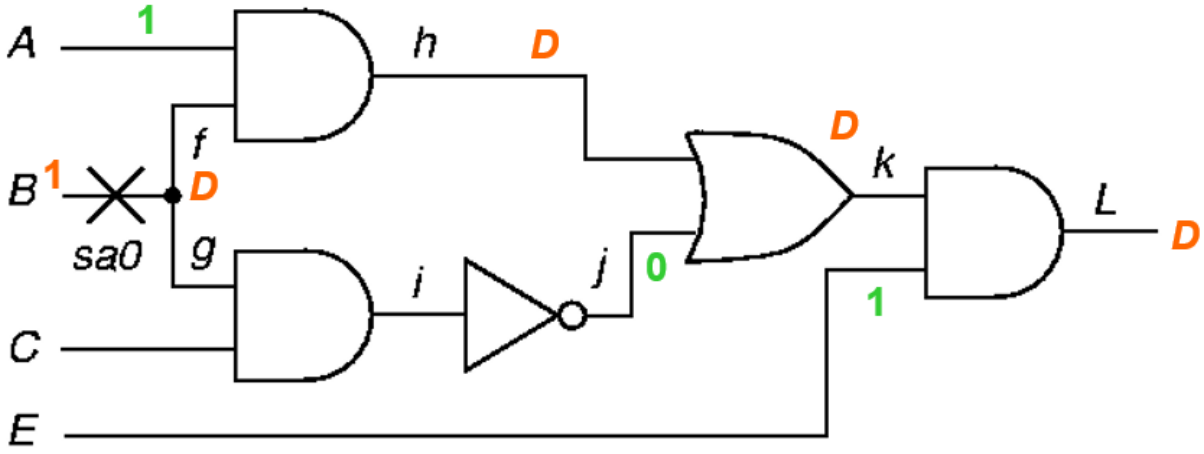


Рисунок 2.3.2 – Перший етап проходження

3. Повернення

Нам потрібно визначити які сигнали подати на решту входів схеми. Для цього починаємо рухатися у зворотньому напрямку (рис. 2.3.3). Проставляємо значення на входи елементів спираючись на таблицю істиності кожного з елементів.

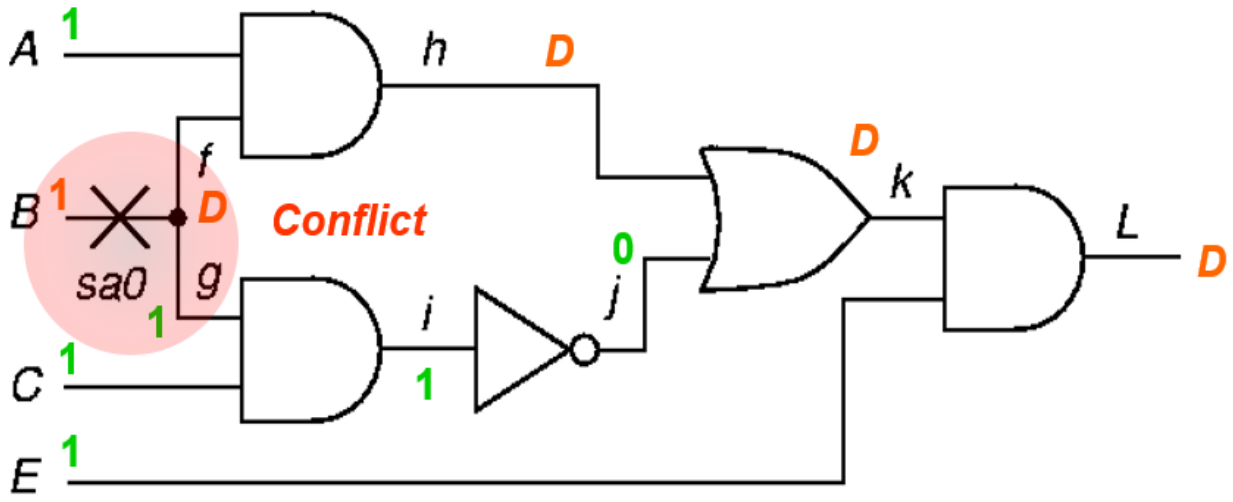


Рисунок 2.3.3 – Другий етап проходження

Через елемент інверсії на провід і попадає одиниця. На елементі AND, щоб отримати на виході 1 потрібно на входи подати також дві логічні 1. Отже на вході E та проводі g маємо логічну 1, що суперечить нашому помилковому 0. Отже цей варіант нам не підходить. Почнемо спочатку й знайдемо інший шлях до виходу із схеми. Про аналізуємо варіант через елементи AND, NOT, OR, AND (рис. 2.3.4).

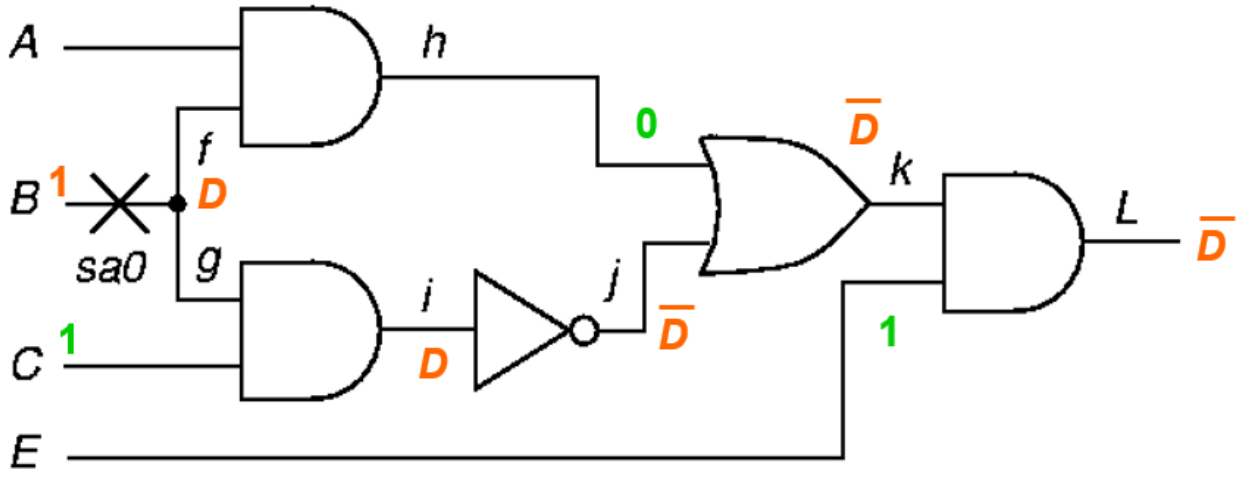


Рисунок 2.3.4 – Перший етап проходження

Аналогічно попередній схемі проходу проставляємо значення на логічних елементах схеми. Спочатку в одному напрямку - по D-кубам й в протилежному – по таблицям істиності (рис. 2.3.5).

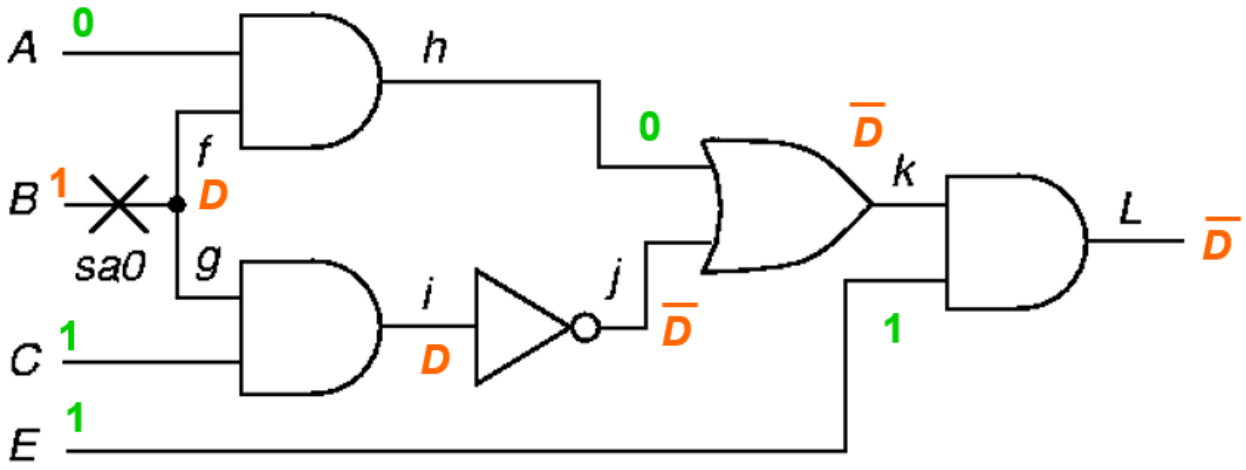


Рисунок 2.3.5 – Другий етап проходження

В результі маємо один тестовий вектор A = 0, B = 1, C = 1, E = 1, який зможе однозначно виявити таку несправність схеми.

2.4 Програмна складова

Щоб реалізувати програмний модуль для знаходження та аналізу тестових наборів комбінаційних схем, було прийнято рішення розробити його на базі web-додатку. У цього способу реалізації є дуже багато плюсів:

- постійний доступ до сервісу (on-line);
- доступ із смартфону, що дозволяє відмовитися від настільних компьютерів на підприємствах. Це дуж сильно зменшує затрати на тестування реальних схем;
- постійне оновлення
- легке користування

Для реалізації користувацького інтерфейсу в якості основної мови програмування було використано JavaScript, оскільки ця мова дозволяє

швидко візуалізувати будь-які контролі без особливих складностей та підтримується усіма сучасними браузерами.

javascript

В даний час Javascript є одним з найпопулярніших мов на ринку. JavaScript стоїть на другому місці. Він в основному використовується при створенні веб-сайтів і веб-додатків.

javaScript - це клієнтський мову, який використовується для створення веб-сторінок. Це окрема мова, розроблена в Netscape. Вона використовується, коли веб-сторінка повинна бути динамічною і додавати спеціальні ефекти на такі сторінки, як перекидання, розгортання і багато типів графіки. Вона підтримує зовнішні програми, такі як документи PDF, що запускають віджети, підтримку додатків флеш-пам'яті і т. д.. Вона також може завантажувати вміст в документ, коли це потрібно користувачеві, навіть без перезавантаження всю сторінку.

Вона взаємодіє з браузером, що не відправляючи повідомлення туди і назад на сервери. JavaScript використовує інтерфейси прикладного програмування (API), які надають додаткові можливості для програміста.

Node JS побудований на середовищі виконання Chrome Javascript для створення швидких і масштабованих мережних додатків. Javascript використовується для обробки HTTP-запитів і генерації вмісту. Коли користувач пише великі програми на JavaScript на клієнті, він може навіть написати логіку на JavaScript на сервері, щоб зробити когнітивні скачки з однієї мови на іншу.

Використовуючи Node JS, можна створити веб-сервер. Переваги Node JS в тому, що він управляється подіями і не буде чекати відповіді на попередній виклик. Він переходить до наступного виклику і використовує події для отримання повідомлень при отриманні відповіді на попередній виклик.[5]

Найважливіша річ, яку можна зробити за допомогою JavaScript, - це створення додатків без веб-контекстів. Мобільні телефони, які в основному

доступні в Apple і Android, використовуються для їх створення на двох різних мовах. Повинна бути можливість написати один раз і використовувати його на обох платформах цих пристроїв. PhoneGap є основою, яка дозволяє це. Також нещодавно у нас з'явився React Native, який служить для цього. Отже, Javascript може використовуватися для розгортання та завантаження відповідних додатків в різних середовищах.

Через усі ці плюсів JavaScript було б абсолютно ясно, що JavaScript - це мова, яка має залишитися. Завдяки всіх функцій розробки інтерфейсів і бекенда JavaScript допомагає підтримувати і те, і інше, а також створювати кращі програми.

JavaScript - один з найпростіших, універсальних і ефективних мов, які використовуються для розширення функціональності веб-сайтів. JavaScript Development Servicesпомогает з легкістю створювати візуальні ефекти на екрані, а також легко обробляти і обчислювати дані на веб-сторінках. Мова програмування також допомагає в розширених функціональних можливостях для веб-сайтів, які використовують сторонні скрипти, серед кількох інших корисних функцій.

Ось деякі ключові переваги JavaScript:

1. JavaScript - це мова клієнта

   Код JavaScript виконується на процесорі користувача, а не на веб-сервері, що економить пропускну здатність і навантаження на веб-сервер.

2. JavaScript - легкий мову для вивчення

   Мова JavaScript простий у вивченні і має синтаксис, близький до англійського. Він використовує модель DOM, яка надає безліч визначених функцій для різних об'єктів на сторінках, що дозволяє без праці розробити сценарій для вирішення призначених для користувача завдань.

3. JavaScript порівняно швидкий для кінцевого користувача

Оскільки код виконується на стороні клієнта, результати і обробка завершуються практично миттєво в залежності від завдання (завдання в JavaScript на веб-сторінках, як правило, прості, щоб не допустити перевантаження пам'яті), оскільки його не потрібно обробляти в веб-сервер сайту і відправляється назад користувачеві, споживаючи як локальну, так і серверну пропускну здатність.

4. Розширена функціональність для веб-сторінок

   Сторонні надбудови, такі як Greasemonkey, дозволяють розробникам JavaScript писати фрагменти JavaScript, які можуть виконуватися на потрібних веб-сторінках для розширення його функціональності (рис. 2.4.1). Якщо ви користуєтеся веб-сайтом і вимагаєте включення певної функції, ви можете написати її самостійно і використовувати надбудову, наприклад Greasemonkey, для її реалізації на веб-сторінці.

5. Компіляція не потрібно

   JavaScript не вимагає процесу компіляції, тому компілятор не потрібно. Браузер інтерпретує JavaScript як теги HTML.

6. Легко налагоджувати і тестувати

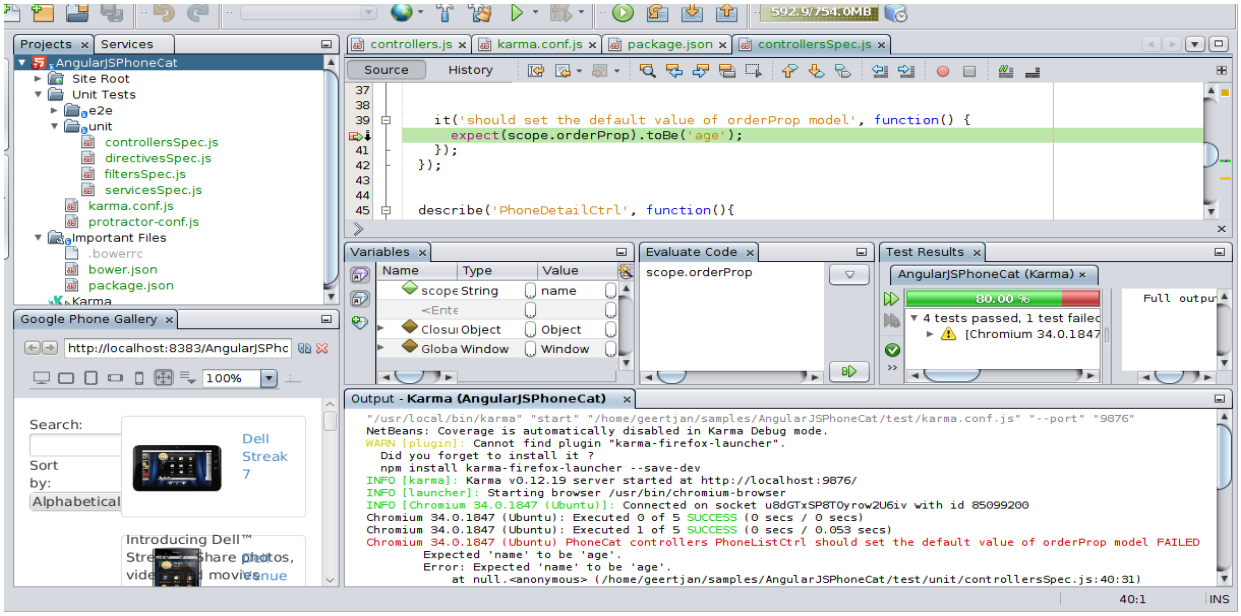


Рисунок 2.4.1 – Інтерфейс JavaScript

Синтаксис розуміння JavaScript простий. Будь-яка людина може вивчити це дуже легко і використовувати його для розробки динамічних і масштабованих сайтів (рис. 2.4.1).

7. Платформа незалежна

   Будь-браузер з підтримкою JavaScript може розуміти і інтерпретувати код JavaScript. Будь-код JavaScript може бути виконаний на різних типах обладнання, для якого написана програма JavaScript.

8. Процедурні можливості програмування

   Мова JavaScript включає в себе всі можливості процедурного мови. Розгалуження, зациклення, перевірка стану - ось деякі з тих можливостей, які можуть бути виконані на веб-сторінці.

Подання елементів схеми реалізовано на базі бібліотеки vis.js.

Vis.js - это динамическая библиотека визуализации на основе браузера. Библиотека разработана для простоты использования, обработки больших объемов динамических данных и обеспечения возможности манипулирования данными.

Библиотека разработана Almende BV . Vis.js отлично работает в Chrome, Firefox, Opera, Safari, IE9 + и в большинстве мобильных браузеров (с полной поддержкой сенсорного ввода).

Для відображення данних було використано HTML та CSS.

HTML є стандартною мовою розмітки для створення веб-сторінок. HTML розшифровується як Hyper Text Markup Language. HTML описує структуру веб-сторінок з використанням розмітки. Елементи HTML є будівельними блоками сторінок HTML. HTML-елементи представлені тегами.

Це означає, що якщо ви хочете розробити веб-сторінку, HTML використовується для створення структури і всього веб-сторінки. HTML для веб-сторінки, як ДНК для тіла.

Проте, все в HTML є стандартним. Наприклад, Тільки Times New Roman використовується для тексту. Звичайно, веб-розробники хочуть, щоб їх веб-сторінки були барвистими і різними. Це де CSS вступає в роботу.

Каскадні таблиці стилів (CSS) - це мова таблиць стилів, використовуваний для опису представлення документа, написаного на HTML або XML (включаючи діалекти XML, такі як SVG або XHTML). CSS описує, як елементи повинні відображатися на екрані, на папері, в мові або на інших носіях.

CSS обробляє розмір шрифту, мову шрифту, фоновий колір, колір переднього плану ... Все!

Без CSS. HTML буде занадто простим і нудним. Без HTML CSS не існувало б.

Якщо ви хочете розробляти або впроваджувати веб-сайти, HTML і CSS незамінні. Вони і JavaScript - три основні технології для всього, що є в мережі. Якщо ви хочете програмувати настільні або чисто мобільні додатки, вони не так важливі. Але більшість програм в даний час доступні через Інтернет.

Також була використана бібліотека jQuery. jQuery - це швидка, невелика і багатофункціональна об'єктна бібліотека JavaScript, яка ознаменувала значний прорив для інтерфейсної веб-розробки, програмування, яке включає в себе створення всього, що користувач бачить і взаємодіє на веб-сайті. Універсально використовувана технологія на стороні клієнта, JavaScript (JS) породила ряд фреймворків і бібліотек, але jQuery був першим, хто торкнувся сумісність браузера з мовою.

Сам по собі JavaScript мав тенденцію діяти трохи по-різному в кожному браузері, вимагаючи від розробників написання коду для конкретного браузера. jQuery може гарантувати, що JS працює однаково в кожному браузері, без необхідності коду, специфічного для браузера. Він також розроблений, щоб спростити використання JavaScript на вашому веб-сайті,

спрощуючи складний код JavaScript в стислі багаторазово використовувані пакети, які дозволяють вам «писати менше, робити більше»

jQuery - це бібліотека об'єктів, яка надає оптимізовану функціональність JavaScript. Сумісність - це перевага jQuery; він дозволяє JavaScript виглядати однаково практично у всіх браузерах без додаткових рядків коду.

Інші переваги:

- Дуже добре підходить для створення користувацьких інтерфейсів, веб-додатків і мобільних сайтів
- Обробляє зовнішні запити і маніпулювання DOM (режим, за допомогою якого JavaScript взаємодіє з HTML), що змушує сайти реагувати на взаємодію з користувачем
- Має простий у використанні API, який значно спрощує анімацію, обробку подій і запити AJAX.
- Доступний і простий у використанні, що дозволяє будь-кому, від інженера до Некодуючі дизайнера, маніпулювати сайтом

Google і Microsoft обидва розміщують jQuery як CDN (мережі доставки контенту), альтернативу завантаження та встановлення jQuery, і, оскільки jQuery, ймовірно, вже кешируєтся на більшості комп'ютерів з інших сайтів, це призводить до більш швидкому часу завантаження.

Як і більшість інтерфейсних технологій, відповіді jQuery засновані на подіях. Кожен раз, коли користувач клацає, відправляє, прокручує, завантажує або виконує будь-яке інше взаємодія, він запускає «подія», і селектори jQuery визначають місце розташування, вибирають і потім маніпулюють елементом HTML з використанням синтаксису CSS. Наприклад, якщо клацання миші повинен змінити колір об'єкта на екрані, jQuery обробляє цей запит, а потім змінює колір цього об'єкта в HTML.

До числа сайтів на основі jQuery відносяться: Google, Microsoft, IBM і Netflix.

Серверна частина продукту розроблена на мові програмування C #.

C # - потужна і гнучка мова програмування. Як і всі мови програмування, вона може використовуватися для створення різних додатків. Ваш потенціал з C# обмежений тільки вашою уявою. Язик не накладає обмежень на те, що ви можете зробити. C# вже використовувався для таких різноманітних проектів, як динамічні веб-сайти, інструменти розробки і навіть компілятори.

C# був створений як мова об'єктно-орієнтованого програмування (ООП). Інші мови програмування включають об'єктно-орієнтовані функції, але далеко не всі повністю об'єктно-орієнтовані.[5]

Чому C #?

Багато хто вважав, що в новій мові програмування немає необхідності. Вважалося, що Java, C++, Perl, Microsoft Visual Basic і інші існуючі мови пропонують всю необхідну функціональність.

C # - це мова, похідний від C і C ++, але вона була створена з нуля. Microsoft почала з того, що працювало в C і C ++, і включила нові функції, які полегшили б використання цих мов. Багато з цих функцій дуже схожі на те, що можна знайти в Java. В кінцевому рахунку, у Microsoft було кілька цілей при створенні мови. Ці цілі можуть бути узагальнені каже Microsoft про C#:

- C# простий.
- C# сучасно.
- C# є об'єктно-орієнтованим.

Крім причин Microsoft, є й інші причини використовувати C#:

- C# потужний і гнучкий.
- C # це мова декількох слів.
- C # є модульним.
- C # буде популярний.
- C # простий

C# усуває деякі складності і пастки мов, таких як Java і C++, включаючи видалення макросів, шаблонів, множинного спадкоємства і віртуальних базових класів. Це все області, які викликають плутанину або потенційні проблеми для розробників C++. Якщо ви вивчаєте C# як свою рідну мову, будьте впевнені - цим темам вам не доведеться виділяти час![5]

C# простий, тому що він заснований на C і C++. Якщо ви знайомі з C і C++ або навіть Java, ви знайдете C# дуже знайомим у багатьох аспектах. Оператори, вирази, оператори та інші функції взяті безпосередньо з C і C++, але поліпшення роблять мову простіше. Деякі з поліпшень включають усунення надмірності. Інші галузі поліпшення включають додаткові синтаксичні зміни. Наприклад, в C++ є три оператора для роботи з членами: ::,. І ->. Знання того, коли використовувати кожен з цих трьох символів, може бути дуже заплутаним в C++.[2] У C# всі вони замінені одним символом - оператором «точка». Для нових програмістів ця і багато інших функцій виключають плутанину.

C# сучасний

Що робить сучасна мова? Такі функції, як обробка виключень, прибирання сміття, розгортаються типи даних і захист коду, - це функції, які очікуються від сучасної мови. C# містить все це.

C# є об'єктно-орієнтованим

Ключами до об'єктно-орієнтованого мови є інкапсуляція, успадкування і поліморфізм. C# підтримує все це. Інкапсуляція - це розміщення функціональності в одному пакеті. Спадкування - це структурований спосіб розширення існуючого коду і функціональності в нові програми і пакети. Поліморфізм - це здатність адаптуватися до того, що необхідно зробити. Це дуже спрощені визначення. Реалізація їх трохи складніше.

С # потужний і гнучкий

Як згадувалося раніше, з C# ви обмежені тільки вашою уявою. Язик не накладає обмежень на те, що можна зробити. C# може використовуватися для таких різноманітних проектів, як створення текстових процесорів, графіки, електронних таблиць і навіть компіляторів для інших мов.

C# - це мова декількох слів

C# - це мова, яка використовує обмежену кількість слів. C# містить тільки декілька термінів, які називаються ключовими словами, які служать основою, на якій будується функціональність мови. Більшість з цих ключових слів використовуються для опису інформації. Ви можете подумати, що мова з великою кількістю ключових слів буде більш потужним. Це не правда При програмуванні на С # ви виявите, що його можна використовувати для виконання будь-якого завдання

C# є модульним

Код C# може бути написаний шматками, званими класами, які містять підпрограми. Ці класи і методи можуть бути повторно використані в інших додатках або програмах. Передаючи частини інформації класів і методів, ви можете створювати корисний, багаторазово використовуваний код.

C# буде популярним

C# є одним з новітніх мов програмування. Невідомо, якою буде популярність С #, але можна посперечатися, що по ряду причин він стане дуже популярною мовою. Одна з ключових причин - Microsoft і обіцянки .NET.[2]

Microsoft .NET є ще однією причиною, чому у C# є шанс домогтися успіху. .NET - це зміна в способі створення та реалізації програм. Хоча з .NET можна використовувати практично будь-яку мову програмування, С # виявляється тією мовою, яку ми вибрали

В підсумку:

C# має такі плюси: простота, об'єктна орієнтація, модульність, гнучкість і стисліcть. Як мова програмування, C# бере найкраще з мов програмування, які вже існують, і переводить їх на нові мови.

C# не складніше у вивченні, ніж C, C++ або Java. Існує безліч книг для вивчення C#. Якщо Microsoft зможе виконати свою обіцянку з .NET, то C # повинен мати довгу і надійну життя, а розробники на C# повинні стати такими ж затребуваними, як програмісти на C++ і Java.[5]

3. ОПИС ПРОГРАМНОГО МОДУЛЯ

3.1 Загальний опис та структура розробленої системи

Розроблювана система призначена для пошуку тестових наборів, що виявляють помилки в комбінаційних схемах. А також для пошуку точного місця знаходження тої чи іншої помилки.

Користувачем цього продукту може бути кожна людина. На практиці це будуть інженери- тестувальники, що будуть працювати на підприємствах, що працюють з великими електронними схемами. Великими плюсом є те, що була обрана ідея для розробки саме web-додатку. Реалізована в такий спосіб система дає змогу тестувальникам на підприємстві аналізувати електронні схеми прямо з телефона або з робочого планшету й не використовувати комп'ютер. Цей факт дає змогу відмовитись від закупівлі дорогих настільних комп'ютерів, що займають багато місця та часу. Раніше доводилося працювати на листочку. Брати великі паперові схеми й з ними йти до реальних величезних схем та роботи якісь помітки, аналіз після чого потрібно було вертатися до робочого місця й робити обчислення й аналіз на настільних комп'ютерах. Завдяки цій розробці робітники мають полегшення роботи та зменшення витраченого час на кожну схему, що призводить до неабиякої економії для власників виробництва або підприємства. Ще великою економією є факт того, що непотрібно купувати настільних комп'ютерів кожному робітнику, що в наш час є дуже затратною частиною вкладень в підприємство. Клієнтами можуть бути власники бізнесу, що працюють з комбінаційними схемами. Це можуть бути чи підприємства, що займаються виробництвом різних схем або підприємства, що працюють з ними, або їх обслуговують. Частіше за все реальними користувачами будуть спеціалісти в тестування та роботі з різними видами схем. Тому проблема з налаштуванням або користуванням даного продукту відпадає сама собою. Вони мають можливість

вивчати задану схему, бачити її таблицю істинності та аналізувати її. Якщо користувачеві потрібно перевірити, чи є в тому чи іншому місці схеми помилка, а саме замикання в одному стані, що свідчить про збій в роботі на попередньому елементі, він просто вводить в програму місце для перевірки, перемичку, й вибирає помилку для перевірки. Через мить тестувальник бачить тестові вектори, що разом називаються тестовим набором, які дають змогу повністю визначити чи є помилка в заданому місці чи все працює в штатному режимі.

Отже, основними діями користувача після відкриття програми є:

- введення нового місця знаходження помилки до системи або продовження роботи зі старими даними;
- введенні помилки;
- аналіз вихідних даних.

Дана схема передбачає наступну роботу проекту: першим кроком є введення даних щодо перевіряємої помилки користувачем, яку необхідно проаналізувати, при успішному введенні даних одразу запускається алгоритм пошуку тестових наборів. Схема передбачає неодноразове вирахування результату, тому є можливість заново запустити алгоритм пошуку тестових векторів при різних введених даних.

Програмна реалізація алгоритмів.

D-алгоритм. У другому розділі вже розповідалось про принцип роботи алгоритму Рота, але коли його потрібно реалізувати програмно це вже інша задача. Треба було взяти до уваги всі можливі компоновки логічних схем з елементів й кожну несправність на елементах. Головним принципом була розробка простого й універсального алгоритму роботи із схемою.

Є три модулі які працюють за для D-алгоритму. Модуль пошуку тестових наборів, модуль проходження по схемі, модуль обробки елементів схеми.

Модуль пошуку тестових наборів є управляючим модулем у цій трійці. Він спочатку працює з одним модулем беручи у нього інформацію про схему та

тип елементів. Постійно потрібно реалізовувати переходи з таблиць істинності до таблиць виродженого покриття а потім ще перетворювати у D-куб.

Швидко дія програмного модуля.

3.2 Інструкція користувача

Заходячи у додаток вас одразу зустрічає дуже інтуїтивний інтерфейс роботи. Для того, щоб знайти тестові вектори тої чи іншої комбінаційної схеми, все, що вам потрібно це задати елемент на якому перевіряється помилка (провід з'єднання) й задати значення константної несправності. Це можна побачити на представленому скріншоті (рис. 3.2.1)

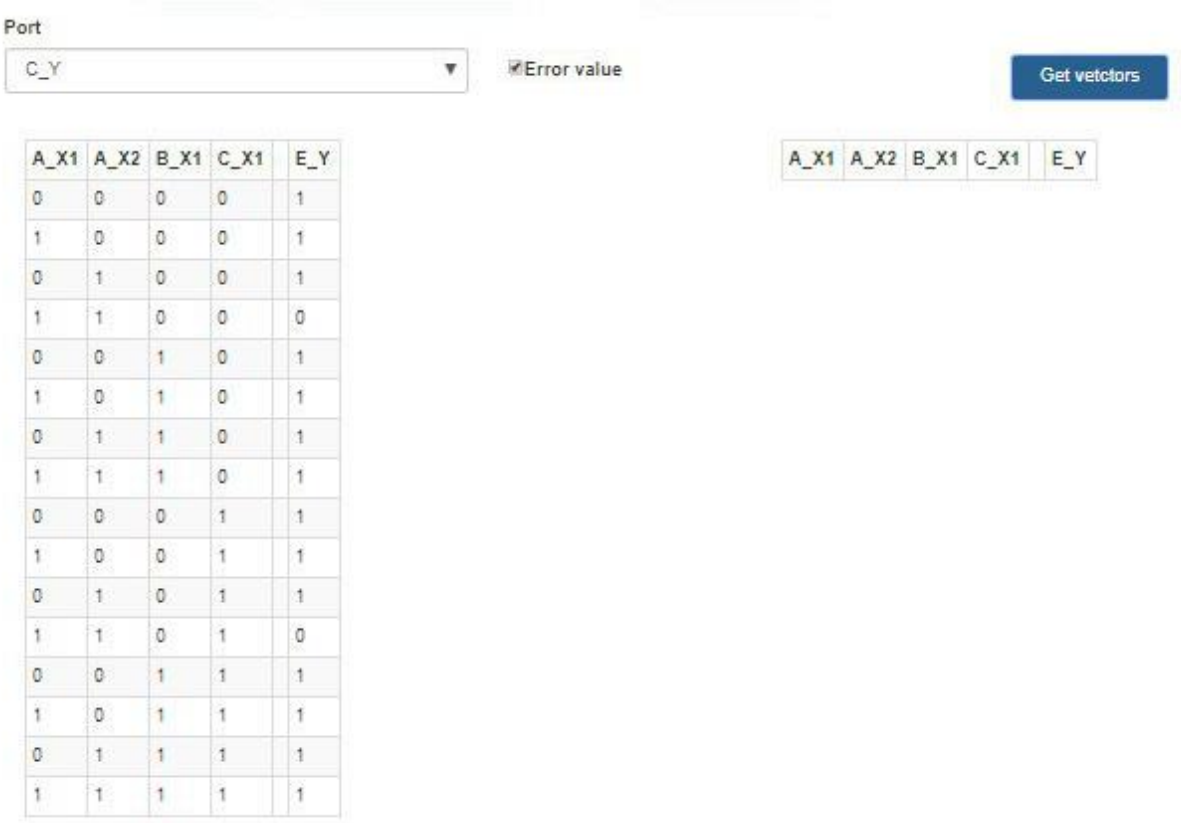


Рисунок 3.2.1 – Інтерфейс розробленної програми

Перше, що бачить користувач – це список пустих полів, що потрібно заповнити. Розглянемо кожне з полів, що необхідні для внесення інформації:

1. Port – поле для введення елемента на якому перевіряється помилка.
2. Error value – поле для введення значення логічного 0. Якщо галочка не стоїть – значить досліджується помилка з постійним «0», але якщо стоїть то з постійною «1».

Коли всі поля заповнені потрібно натиснути кнопку «Get vectors». Після чого програма виконує пошук всіх потрібних тестових векторів для виявлення заданої несправності, враховуючи всі задані користувацькі параметри. Потім виводиться інформаційне повідомлення з результатами (таб. 3.2.1), де представлені усі тестові вектори для даної комбінаційної схеми. Цих векторів цілком і повністю достатньо для того, щоб, призначивши такі вхідні сигнали, точно знати про помилку й її місце знаходження. Кожен рядок цієї таблиці відповідає наступному вектору для тестування схеми.

Зробивши один крок назад можна повернутися до вводу даних й зробити нову побуду векторів для нового місця поломки.

A_X1	B_X1	B_X2	C_X2		D_Y
1	0	0	1		0
1	1	0	1		1
0	0	1	1		0
1	0	1	1		1
0	1	1	1		0
1	1	1	1		1

Таблиця 3.2.1 – Таблиця результатів

Для тестування нашої комбінаційної схеми, потріно взяти перший знайдений вектор й подати на входи схеми, що випробується, ті сигнали, що зазначені в вихідній таблиці (табл. 3.2.1). Після подачи потрібних сигналів спостерігаємо за чигналами, що прийдуть на вихід із схеми. Потім починаемо порівнювати сигнали, що прийшли на вихід з тими, що ми мали отримати (вони зазначені в тестових наборах). Якщо хочаб один з виходів не співпав з

очікуваним результатом, це свідчить про сто відсоткову присутність помилки. Такі кроки потрібно провести з усма заданими векторами, щоб бути певним, що помилка присутня й присутня саме в вказаному місці.

3.3 Перевірка правильності формування тестових векторів

Проведемо аналіз й пошук тестових векторів наступної схеми. Спочатку скористаймося розробленою програмою й знайдемо тестові вектори. Задаємо схему (рис. 3.3.1) в програму.

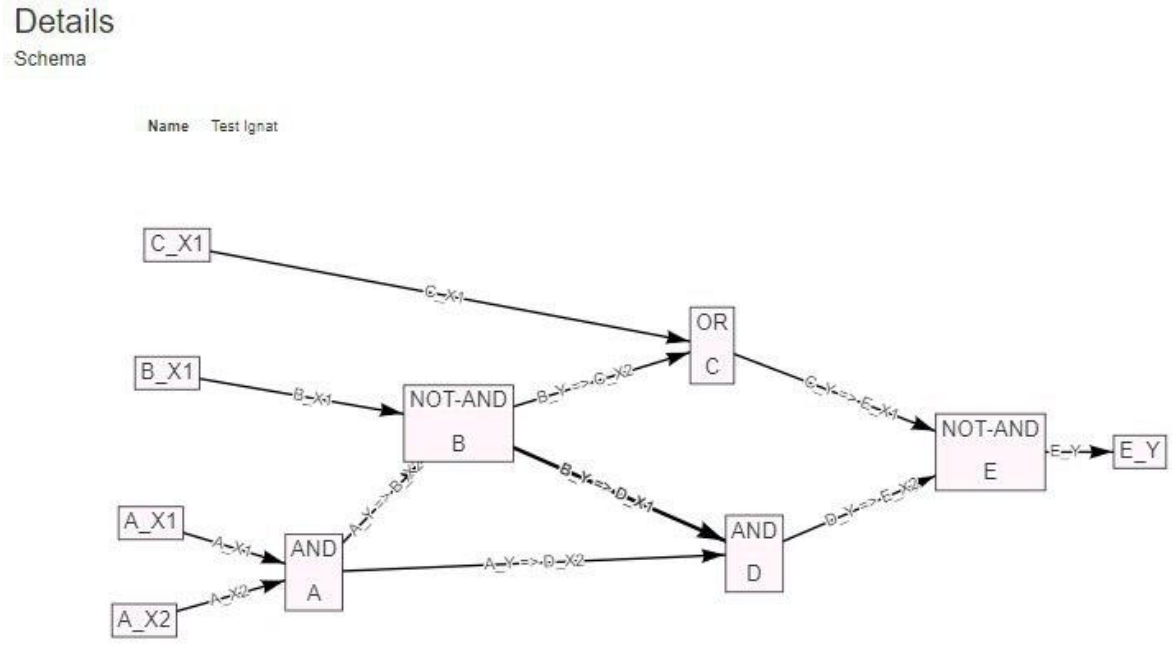
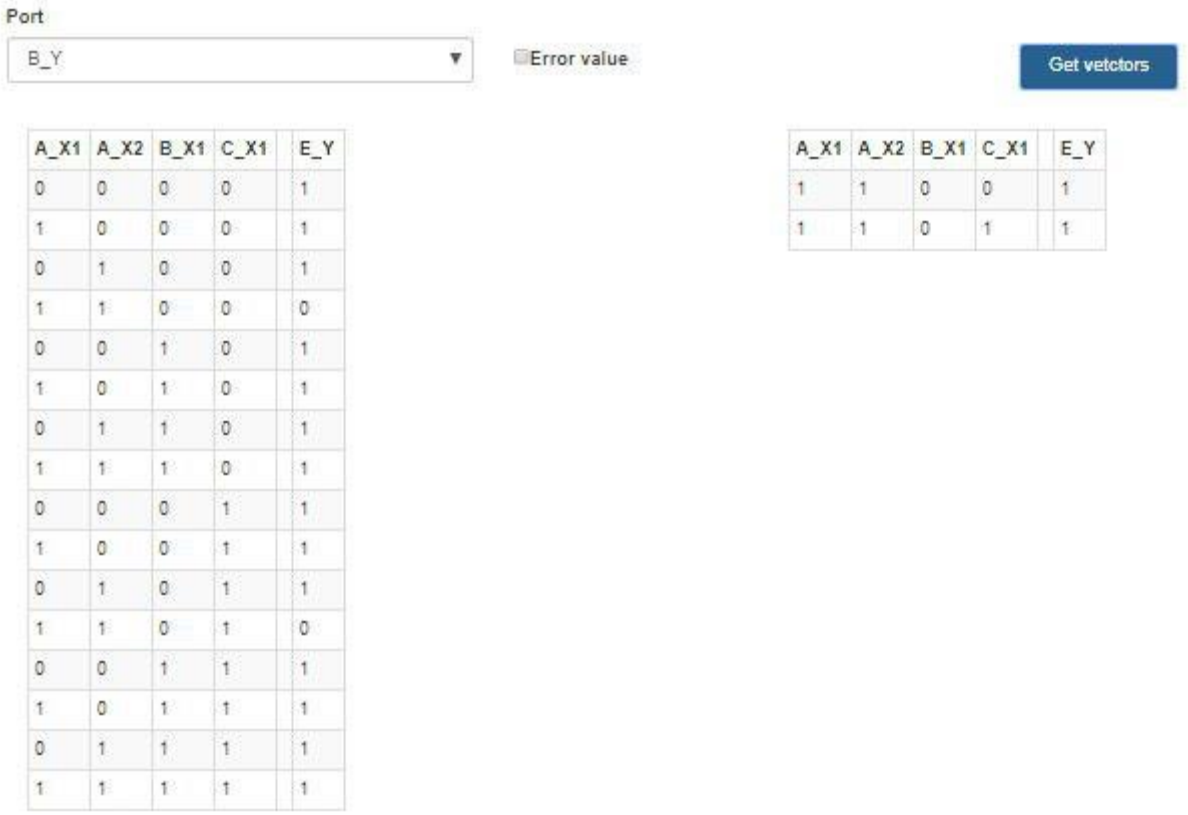


Рисунок 3.3.1 – Комбінаційна схема

В поле для вводу місця помилки вводимо потрібний нам, для дослідженя, участок зв'язку по якому йде помилка. В нашому випадку це вихід з елементу B_Y як показано на скриншоті програми (рис. 3.3.2). Ми досліджуємо помилку постіного «0» на виході з елемента B. Тому не ставимо галочку в полі «Error value». Після чого натискаємо кнопку «Get vectors» й бачимо таблицю з правої сторони екрану в які вказані тестові вектори. Кожний рядок є унікальним тестовим аектором. Отже якщо ми подамо на входи схеми такі данні:

- На A_X1 обов’язкова «1»;
- На A_X2 обов’язкова «1»;
- На B_X1 обов’язковий «0»;
- На C_X1 будь-яке значення.

А на виходах не отримаємо одиниць, то стовідсоткаво маємо постінй помилку «0» на виході з елемента B.



A_X1	A_X2	B_X1	C_X1	E_Y
0	0	0	0	1
1	0	0	0	1
0	1	0	0	1
1	1	0	0	0
0	0	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1
0	0	0	1	1
1	0	0	1	1
0	1	0	1	1
1	1	0	1	0
0	0	1	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

A_X1	A_X2	B_X1	C_X1	E_Y
1	1	0	0	1
1	1	0	1	1

Рисунок 3.3.2 – Інтерфейс програми

Разом усі тестові вектори називаються тестовим набором.

Вирішемо цю задачу без програмних помічників. Намалюємо схему (рис. 3.3.3). Позначемо червоним крестиком місце помилки й вкажемо занчення її «0». Починаємо прохід в сторону виходу. По таблиці D-кубів визначаємо вихід з елементу B и це D. Наступним елементом D є елемент AND. Нам потрібно на вихід, щоб прошла D, тому по таблиці кубів на другій

вхід елемента маємо подати «1». Просуваємося далі й зустрічаємо елемент NOT- AND. Таким самим чином встановлюємо занчення на другий вхід так, щоб на вихід пішла D. Дійшли до виходу – просування завершено.

Починаємо довизначення елементів спираючись на таблиці істинності кожного зних. По другому входу елемента E рухаймося на елемент C. По таблиці істнності візначаємо, що щоб отримати одиниц на елементі «OR» потрібно подати щось крім двох нулів.

Повертаймося до елемента D й рухаймося на елемент A. Потаблиці істинності визначаємо, що щоб отримати одиницю на елемнті AND нам потрібні тільки дві одиниці.

Після цього етапу маємо завершений пошук тестових векторів, а саме:

- На A_X1 обов’язкова «1»;
- На A_X2 обов’язкова «1»;
- На B_X1 обов’язковий «0»;
- На C_X1 будь-яке значення.

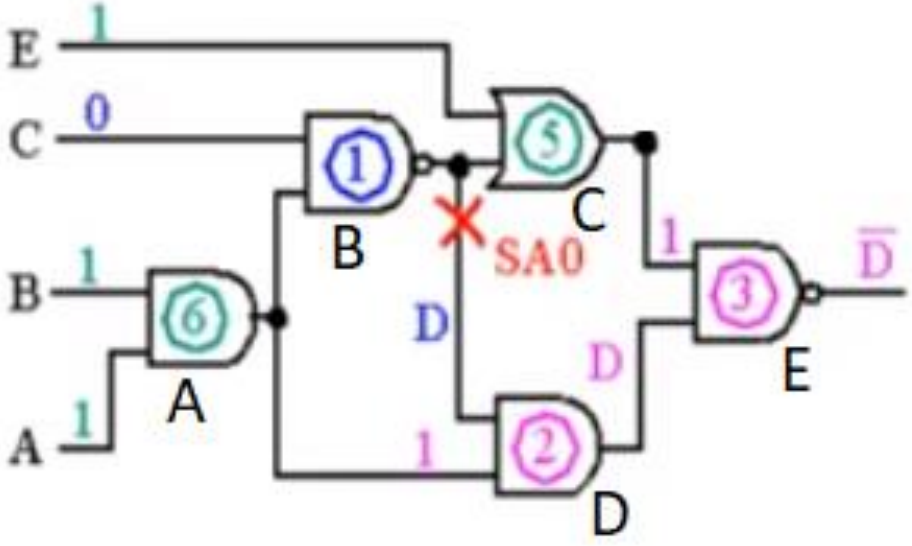


Рисунок 3.3.3 – Логічна схема

Отже маємо повністю робочий програмний продукт, що абсолютно коректно видає тестові набори за лічені секунди.

3.4 Опис розроблених модулів

Для реалізації функціоналу зазначеного вище було створено наступні модулі:

- Модуль управління. Модуль, що відповідає за взаємо зв’язок з всіма модулями та розподіляє всі поставлені завдання. Через цей модуль проводяться всі запити та керування системою.
- Модуль комутації. Цей модуль відповідає за взаємо зв’язок між моєю програмою та програмою, що формує мінімізовані комбінаційні схеми з двовходовими елементами. Ця під програма видає на вихід таблицю істинності і які саме логічні елементи присутні в схемі.
- Модуль обробки вхідних даних. Модуль, що оброблює вхідні данні, що отримала програма від попередньої програми. Аналізуючи таблицю істинності, перевіряє правильність та коректність отриманих даних.
- Модуль типізації даних. Цей модуль відповідає за перетворення рядків таблиці істинності в вектори. А також для перетворення таблиць виродженого покриття в таблиці D-кубів.
- Модуль обробки таблиці істинності. Цей модуль зберігає усі потрібні таблиці істинності.
- Модуль пошуку тестових наборів. Цей модуль відповідає за взаємозв’язок між модулями обробки елементів схеми та модулем проходження по схемі та розподіляє всі поставлені завдання. Саме вони працюють з пошуком тестових наборів з обробкою даних векторів з таблиці істинності й таблиць виродження й векторів з D-

кубів. Саме з цього модуля програмного продукту отримуємо вихідними даними бажані тестові набори.

- Модуль обробки елементів схеми. Задача цього модуля це робота безпосередня з елементами схеми та створення таблиць істинності для кожного елемента схеми. Після чого йде перетворення таблиці виродженого покриття в таблицю D-кубів відповідно. Також даний модуль видає значення з потрібної таблиці для розрахунків модулю пошуку тестових наборів, що робить виклик до нього.
- Модуль проходження по схемі. Цей модуль працює безпосередньо з обробкою кожного елемента в заданій схемі спираючись на усі таблиці, що оброблює модуль обробки елементів цієї схеми. На першому етапі, при проходженні з входів схеми до виходів (режим активації), йде робота з таблицями D-кубів. Програма аналізує наступний елемент та спираючись на відповідну таблицю визначає на присвоює значення сигналу на вільні виходи та входи логічних елементів, запам’ятовуючи їх. Після ініціалізації цього шляху (від заданого місця помилки до одного із виходів) починається ініціалізація в зворотному порядку. На цьому шляху йде присвоєння значень сигналів за таблицями виродженого покриття.

3.5 Перспективи розвитку програмного продукту.

На сьогоднішній день розроблено головну структуру цього великого проекту. В майбутньому планується дуже багато різних покращень, що дадуть змогу зробити роботу та аналіз схем більш простим та дешевим. Планується розробити програмний модуль, що зможе знаходити тестові вектори не тільки для комбінаційних схем з двовходовими елементами, але й для інших електричних схем з великою кількість складних елементів.

Також планується розширити функціонал самого додатку. Користувач матиме змогу регіструватися та мати свій особистий кабінет. Щоб бцло

можливість зберігати та відновлювати свої тести . Робити план на наступний робочий день або тиждень. Можливість зберігати результати тестів в окремі файли та потів відновлювати їх звідти.

ВИСНОВОКИ

У даній роботі було проаналізовано методи виявлення несправностей у комбінаційних схемах. Розглянуто методи від найосновніших до останніх швидких інноваційних. Проаналізовано існуючи програмні продукти для роботи з комбінаційними схемами. Нажаль, на сьогоднішній день існує зовсім не багато програмного забезпечення яке знаходиться у відкритому доступі и дає можливість швидкого формування тестів для комбінаційних схем. Є програми які виконують якісні тестування але, щоб мати доступ хоча б до половини функціоналу потрібно заплатити значну суму. У цій роботі реалізовано програмний модуль, яким можна доповнити кожен з проектів, працюючих з моделюванням або аналізом комбінаційних схем. Були взято до уваги та виправлено основні мінуси існуючих рішень та збережені позитивні моменти:

- доступ з будь-якої системи (Windows, Linux, Mac OS, Windows Phone 8, Android, iOS);
- в будь-який час (режим on-line);
- безкоштовний доступ;
- постійна підтримка;
- інтуїтивне користування.

В майбутньому планується додати ще й інші алгоритми пошуку тестових векторів які зможуть працювати з більш складними схемами. Реалізуємо особистий кабінет у web-додатку, що дозволить зберігати та надсилати свої результати роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Моделирование. Тестирование, надёжность, контроль и диагностика компьютерных систем [Електронний ресурс]: учебное пособие для изучения дисциплин «Моделирование» и «Тестирование, надёжность, контроль и диагностика компьютерных систем» для иностранных студентов специальности «Компьютерные системы и сети», «Системное программирование» и «Специализированные компьютерные системы» / НТУУ «КПІ» ; сост. В. В. Гроль, В. А. Романкевич, Е. Р. Потапова.– Електронні текстові дані (1 файл: 782.5 Кбайт). – Киев: НТУУ «КПИ», 2011
2. Гроль В.В., Романкевич В.О. Базові поняття і конструкції мови програмування Сі. Методичні вказівки до вивчення дисципліни “Моделювання” // Київ.– “Політехніка”, 2003.– 24с.
3. Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю. С. , Пиневич М.М. Прикладная теория цифровых автоматов. – К.: Вища Школа. – 1987г. – 375с.
4. Гроль В.В., Романкевич В.А., Потапова Е.Р. Организация процедур логического моделирования цифровых блоков. Методические указания к изучению дисциплин «Моделирование», «Тестирование, надёжность, контроль и диагностика компьютерных систем» // Київ.– “Принт-центр”, 2007.– 44с.
5. Офіційний сайт документації до продукції компанії Microsoft [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Редмонд – Режим доступу: https://docs.microsoft.com (дата звернення 24.05.2019) – Technical documentation, API, and code examples.
6. «Fault Detection and Test Minimization Methods for Combinational Circuits» - SM. Thamarai[#1] , Dr K.Kuppusamy[*2], Dr T.Meyyappan.
7. Michael L. Bushnell, Vishwani D. Agrawal , “Combinational Circuit Test generation” in Essential Of Electronic Testing For Digital Memory and mixed

signal VLSI Circuits. Kluwer Academic Publisher New York, Boston, Dordrecht, London, Moscow.

8. Miron Abeamovici, Melvin A.Breuer, Arthur D.Friedman, “Testing For Single Stuck At Faults” in Digital Systems and Testable Design. IEEE press New York.
9. Карибский В.В., Пархоменко П.П., Согомонян Е.С. Основы технической диагностики / Под ред. П.П.Пархоменко. -М.:Энергия,1976, -463с.
10. Sellers F.F., Hsiao M.Y., Bearnson L.W. Analyzing Errors with the Boolean Difference. -IEEE Trans. on Comput., 1986, v.C-17, p.676-683.
11. Уткин А.А. Универсальный подход к построению тестов.-Сборник научных трудов ИТК АН БССР, Минск, 1981, с.5-16.
12. F. Kocan, and D. Saab, “Concurrent d-algorithm on reconfigurable hardware, “Proceedings of the 1999 IEEE/ACM international conference on Computeraided design, pp.152-156, 1999.